A Flexible Approach to Quantifying Various Dimensions of Environmental Complexity

Michael L. Anderson
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742, USA.
Tel: (301) 405-1746 Fax: (301) 314-1353
Email: anderson@cs.umd.edu

ABSTRACT— In this paper I propose a flexible method of quantifying various dimensions of the complexity of a test environment, including its information density, variability, volatility, inconsistency, and uncertainty. This allows one to determine the task performance of intelligent agents as a function of such measures, and therefore permits derivative measures of their perturbation tolerance—that is, their ability to cope with a complex and changing environment.

Keywords:
complexity, perturbation tolerance, standards, testbeds

1. INTRODUCTION

There are many important and valid approaches to measuring the performance of intelligent systems: provable optimality; typical (and worst-case) solution times; speed and throughput, to name just a few. However, if one is interested, as my research group is, in complex systems operating in changing environments with time and resource constraints—that is, environments where there may *be* no optimal solution, or no time to calculate one, and where experience may confound expectations in significant ways, and thus where finding *some* way to accomplish something is more important than finding the theoretically best way—then such standard measures of performance may not capture the most interesting and crucial elements of performance.

As is well known, maintaining adequate performance in complex and changing environments has been a perennial stumbling-block for intelligent systems, and an ongoing challenge to AI. A typical AI system designed for a specific task often fails utterly when circumstances take it even slightly outside its task specifications. Thus, the ability to handle unexpected difficulties, even if non-optimally—that is, the ability to muddle through difficult situations without breaking down—seems to us an ability worth specific study, and, if possible, implementation.

We call this general ability to cope with a complex and changing environment "perturbation tolerance". The term is meant as an extension and generalization of John McCarthy's notion of "elaboration tolerance"—a measure of the ease with which a reasoning agent can add and delete axioms from its knowledge base [1]. However, our term is more general than McCarthy's because his is explicitly limited to formal, symbolic systems, and an elaboration is defined as an action taken to change such a system [2]. But since a given intelligent agent may well consist of more than just a formal reasoning system, and flexibly coping with a changing world may therefore involve altering components in addition to, or instead of, its formal reasoner, we define a perturbation as any change, whether in the world or in the system itself, that impacts the performance of the agent. Performance is meant to be construed broadly to encompass any measurable aspect of the agent's operation, although, as will be explained below, we tend to favor measures for such things as average reward and percentage task completion over such things as reasoning speed or throughput. *Perturbation tolerance*, then, is the ability of an agent to quickly recover—that is, to re-establish desired/expected performance levels—after a perturbation.

However, if improving perturbation tolerance is to be among the goals for intelligent agents, it will be necessary to quantify and measure this aspect of performance. And if perturbation tolerance is primarily a matter of maintaining performance in the face of various kinds of complexity and change, then it is just such complexity and change that should be the focus of measurement. Further, it would be best if, instead of each lab and working group devising their own set of standards, there were a common standard, and preferably one that might be applied to say something useful about current testbeds. For instance, is Phoenix [3], [4] more or less complex than Tileworld [5]? And how can one compare the relative complexity of two different Tileworld runs?

To this end, I suggest a way to specify an environment that allows for such factors as its complexity, informa-

tion density, variability, volatility and uncertainty to be measured. From such measures I show how derivative measures of environmentally-relative task difficulty and degree of perturbation can be developed, and suggest some different metrics for measuring task performance.

2. COMPARISON WITH RELATED WORK

First, as mentioned already above, it should be made clear that while the approach defined here *could* be used to build new testbeds, it can *also* be used to characterize the properties of existing testbed environments in a uniform way. Thus, what is offered here is less a blueprint for new standard testbed implementations, and more a suggestion for a standard way of measuring some important properties of the testbed environments within which intelligent agents operate. It is perhaps worth noting that the lack of a standard way to evaluate intelligent agents has prompted DARPA to modify their Cognitive Information Processing Technology research initiative to include Cognitive Systems Evaluation as a focal challenge.[1]

One weakness of some domain specifications, from the standpoint of evaluating perturbation tolerance, is that they focus on controlling the characteristics and interactions of the agents *in* the world, rather than on fine control of the world itself. In MICE, for instance [6], the main goal was "an experimental testbed that does not simulate any specific application domain, but can instead be modified to impose a variety of constraints on how agents act and interact so that we can emulate the different coordination issues that arise in various application domains." This strategy is, of course, perfectly sensible when it is the *coordination strategies* of multi-agent systems that is under investigation, but it provides little foundation for measures of perturbation tolerance *per se*.

Another weakness of some domain specifications is the limited number of environmental features that can be easily isolated and measured. For instance, the Phoenix testbed [3], [4] offers ways of building complex and dynamic environments (in which the main task is fighting forest fires), but does not offer a general method for measuring the complexity and dynamicity of those environments. Even what is perhaps the most popular and adjustable of the standard test domains for simulated autonomous agents, Tileworld [5], suffers somewhat from this defect. The main task in Tileworld is to fill holes with tiles, quickly and efficiently, while avoiding obstacles. Among the strengths of Tileworld is its ability to easily measure the performance trade-off between deliberation and reactivity. Tileworld allows one to set the value of such environmental variables as the frequency with which objects appear and disappear, the number and distribution of objects, and the reward value for filling each hole. However, as important as these environmental variables are, there are also other aspects of an environment with which an intelligent agent must cope, and against which performance should be measured. In addition, it is not clear how to translate the variables governing Tileworld to those governing other environments. Finally, Tileworld tests only planning (and plan implementation) performance. But intelligent agents may also need to be able to perform such tasks as the inference-based categorization or identification of objects; the communication of accurate information about an environment; and the mapping of stable environmental features. The current proposal, in providing a more general approach to measuring environmental complexity, aims to lay a foundation for measuring performance in these tasks as a function of the complexity of the environment, and to make cross-domain and even cross-task comparisons easier.

[1] http://www.darpa.mil/baa/baa02-21mod6.htm

3. COMPLEXITY METRICS

It is proposed that the environment be modeled as an n-dimensional grid[2] with a large number of propositions (including sets of numeric values and node activations, to simulate the operation of perceptual NNs, sonar, etc.) that can characterize each location, or "square", in the grid. Each square may be adjacent to (accessible from) one or more other squares. Each proposition p might or might not hold in each square s. As s comes into the perceptual range of the agent, it "picks up" on the propositions that characterize it (propositions consisting of numeric values "stimulate" the appropriate perceptual systems directly; symbolic propositions are entered directly into the agent's knowledge base (KB), and might be thought of as the sort of structured representations that would typically be delivered to an intelligent system by a complex perceptual system like vision).[3] The combination of a grid of a certain size and shape with its characterizing propositions is called an *overlay* (O).

Any given environment has many different features that determine its complexity, independent of the task to be performed in that environment. Specifying the environment in the terms given above allows one to measure these features as follows.

3.1. Basic Measures

[2] For a discussion of the wide applicability of this model, see the subsection on Generality and Extensibility, below.

[3] It is perhaps worth emphasizing that the only propositions relevant to the specification are those characterizing features of the environment that the agent would be expected to perceive or otherwise pick up. The number of water atoms at a given location would not be a relevant proposition unless the agent in question is capable of seeing and counting water atoms. Note the implication that the more perceptually sophisticated the agent, the richer its domain.

n (overlay size): the number of squares in the overlay. If the number of squares changes during the course of an experiment, this will naturally have to be reflected in the measure; whether it is best to use the average size, the final size, or some other measure may depend on the details of the experiment. Note that to choose the number of squares for an environment is also to choose the spatial granularity of the environment. There can be some hidden difficulties here, for instance in the case where different tasks, e.g. navigating a hallway or picking a lock, require that an agent divide space more or less finely. It may the that in these cases, it will be best to treat moving from one task to another in terms of moving from a low-granularity overlay to a high-granularity one.

ρ_I (information density): the average number of propositions characterizing each square.

V_o (variability): a measure of the degree of difference in the characterizing propositions from square to square. V_o can be calculated as the sum of the propositional difference between each pair of squares in the overlay divided by their geometric (minimum graph) distance:

$$V_o = \sum_{i,j=1}^{n} \frac{D_p(s_i, s_j)}{G(s_i, s_j)} \tag{1}$$

Where $D_p(s_i, s_j)$ is the number of propositions that hold in s_i but not in s_j and vice-versa; $G(s_i, s_j)$ is the distance between the squares and n is the total number of squares in the overlay.

δ_o (volatility): a measure of the amount of change in the overlay as a function of time. δ_o can be measured in a way similar to V_o, except that rather than measure the propositional difference as a function of geographical distance, we measure it as a function of temporal distance.

$$\delta_o = \sum_{i,j=1}^{n,t} \frac{D_p(s_{i,1}, s_{i,j})}{j} \tag{2}$$

Where $D_p(s_{i,1}, s_{i,j})$ is the number of propositions that hold in s_i at time 1, but not in s_i at time j, and vice-versa; t is the total time of the simulation, and n is the number of squares in the overlay.

I (inconsistency): the amount of direct contradiction between the beliefs of an agent (in its KB) and the propositions characterizing the environment. Note this must be a measure of the number of *direct* contradictions between p and $\neg p$, since the inconsistency of any two sets of propositions is in general undecidable [7].[4] I can be measured as the percentage of propositions initially in the overlay that directly contradict elements of the agent's initial KB (e.g., 2%, 5%, 10%, 15%, 25%). In the case where $\delta_o > 0$, a more accurate measure might be the *average* percentage of propositions, over time, that directly contradict elements of the initial KB. Note, however, that this measure should *not* reflect the percentage of direct contradiction between the environment over time *and the KB over time*. I is meant to be a measure of one kind of difficulty an agent might face in its environment, that it needs to overcome (or at least manage) in order to successfully cope with that environment. Thus, only the *initial* KB should be used to determine I, for if, through the efforts of the agent, I approaches zero as the test run proceeds, this is a measure of the success of the agent, and does not represent a reduction of the difficulty of the task the agent faced.

U (uncertainty): a measure of the difficulty of perceiving the contents of the square correctly. Uncertainty can be understood as the ratio of the average number of "false" propositions (p_f) in each square to the average total number of propositions in each square.

$$U = p_f / \rho_I \tag{3}$$

In the case where one is building a testbed, uncertainty requires the designer to seed the squares with false or inapplicable propositions, or perhaps, after assigning propositions to each square, to replace a certain number of them with their negations. In the case of modeling an existing testbed, or where using numeric values rather than propositions, if the percentage of time that the system will make perceptual errors is known, this number can be used here.

D_o (overlay difference): a measure of the propositional difference between two overlays O_1 and O_2. D_o can be measured as the sum of the propositional differences between the corresponding squares of each overlay.

$$D_o = \sum_{i=1}^{n} (s_{O_1,i}, s_{O_2,i}) \tag{4}$$

Two overlays may have precisely the same information density, variability and volatility, and still be charac-

[4] A practical aside: work with Active Logic shows that although an indirect contradiction may lurk undetected in the knowledge base, it may be sufficient for many purposes to deal only with direct contradictions. After all, a real agent has no choice but to reason only with whatever it has been able to come up with *so far*, rather than with implicit but not yet performed inferences. Active Logic systems have been developed that can detect, quarantine, and in some cases automatically resolve contradictions [8]–[13].

terized by different propositions; hence this measure of overlay difference. This is useful for cases where an agent is to be trained in one overlay, and tested in another, and the question is how much the differences in the test and target domains affect performance.

It is not expected that every testbed, nor every test run, will make use of all these measures of environmental complexity. Depending on the capabilities of the testbed, and on what is being tested at the time, only a few of these measures may be appropriate. Note further that, depending on the task, some of these measures can simulate others. For instance, even in a completely stable environment ($\delta_o = 0$), the agent can experience the *equivalent* of volatility if $V_o > 0$, for as it traverses the environment each square will offer different information. This difference may not affect the agent at all if its sole task is to map the environment, but it could make an inference-based task more difficult in the same way that a changing environment would. Likewise for the case where $I > 0$, for as the agent encounters these contradictions, they can offer the *equivalent* of change, since change can be understood in terms of p being true at one time, and not true at another. Naturally, determining what manner of variation affects what tasks, and by how much, is one of the items of empirical interest to AI scientists. Isolating these different kinds of complexity and change can help make these determinations more specific and accurate.

3.2. Derivative Measures

The basic measures discussed above can be combined in various ways to construct any number of derivative measures. One such measure of particular importance is of the overall complexity of the environment.

C (complexity): a measure of the overall complexity of the environment. C can be defined as the product of all the non-zero basic measures:

$$C = n \times \rho_I \times V_o \times \delta_o \times (I + 1) \times 100U \qquad (5)$$

The intuition behind this compound measure of complexity is that there are in fact many different reasons that an environment might be difficult to cope with, all of which, therefore, can be considered to contribute in some way to the overall complexity of the environment itself, or to a measure of the environment's contribution to the difficulty of tasks to be performed there. For instance, a large environment is in some sense more complex than a small one *ceteris paribus*, just because there is more of it to deal with. After all, mapping or locating objects in a large environment is likely to be *harder* than doing it in a small one. Likewise, information density captures the notion that a more *intricate* environment—one that requires a greater number of propositions to describe—will be harder to reason about or deal with than a less intricate one. Sometimes this will mean that an intelligent agent has more to think *about* in trying to act in a more intricate environment, and sometimes this will mean it has more to *ignore*; both can be difficult. The variability and volatility of an environment expresses the intuition that an environment that remains more or less the same from place to place, and from time to time, is simpler than one that does not. Inconsistency expresses the idea that an environment that is very different from one's expectations will be harder to deal with than one that is not, and, similarly, uncertainty captures the fact that if it is harder (for whatever reason) to correctly perceive an environment, then certainly coping with it will also be more difficult. The overlay difference allows one to quantify the notion that moving between different domains can be difficult (and is likely to be more difficult as a function of the difference).

It may well turn out, after further consideration, both that there are *more* factors important to the complexity of an environment, and that each factor contributes to a measurably different *degree* to overall complexity (something that might be expressed by adding various coefficients to equation 5). Likewise, perhaps it will turn out that more accurate expression of overall complexity results from *adding* rather than *multiplying* all or some of the various factors. I would welcome such future developments as improvements of the preliminary suggestions I am offering here. Ultimately, an evaluation of the usefulness of these measures will require, and suggestions for improvement will certainly result from, their attempted application in evaluating the performance of intelligent agents in increasingly complex environments. My hope is only that they are well-specified enough in their current form to lend themselves to such use.

3.3. Generality and Extensibility

I have characterized the test environment in terms of a grid of squares of a certain size and shape. Naturally, such a characterization is most directly applicable to artificial environments in fact composed of such a grid ("grid worlds"). However, it should be noted that whenever it is possible to divide a domain into parts, and characterize (the contents of) those parts in terms of some set of propositions, in the sense defined above, then it should therefore be possible to characterize and measure the complexity of that domain in the terms set forth here. We might call such domains "grid-available".

One obvious case of a grid-available domain is one consisting of a mappable terrain (or space) with discrete, localizable features. There are very many domains of this sort, including those, like the world itself, that are not naturally structured according to a grid, i.e. that are

continuous. It is nevertheless possible, albeit with some abstraction, to usefully divide such a domain into spatial parts, and characterize the features of each part in terms of a set of propositions.

Another class of domains that are grid-available are those that, while not strictly-speaking spatial, nevertheless consist of individualizable information-parts. A database is one such domain, and the World Wide Web is another. In each case, the domain consists of individual parts (records, pages), with specifiable contents, that may be adjacent to (linked to, accessible from) one or more other part(s). Depending on the needs of the experiment, an "overlay" might be defined as an entire database or set of web-pages, or some particular subset, as for instance the recordset returned by a given query.

Finally, well-specified state spaces are also grid-available domains. Each state corresponds to a "square" in the grid, and the agent can take actions that move it between states. The states themselves can be characterized in terms of some set of propositions.

Examples of domains that are not grid-available include truly continuous or holistic domains that cannot be usefully broken into parts and/or have few or no local properties (all properties are properties of the whole). Domains described at the quantum level appear to be of this sort, as global quantum properties are often not determined by local ones, making analysis of the parts far less useful than in classically described domains.

4. SAMPLE PERFORMANCE METRICS

In keeping with the philosophy that *flexibility* and *adaptability*—an ability to get along even in difficult circumstances—are among the paramount virtues of cognitive agents, we suggest that evaluating task performance is more important than evaluating such things as reasoning speed, throughput, or the degree of consistency in a post-test KB. Indeed, for an intelligent agent it may be that maintaining a consistent database is in general less important than being able to deal effectively with contradictions while continuing to operate in a dynamic environment.[5] Consider, for instance, a target location task, where the agent must traverse an environment containing 100 targets (lost hikers, for instance) and find them all as quickly as possible. A simple measure of performance here might be:

$$M = \frac{(TP)}{(tA)} \tag{6}$$

[5]This is because, for any sufficiently complex knowledge base that was not produced by logical rules from a database known to be consistent, and/or to which non-entailed facts are to be added (e.g. from sensory information), it will not be possible to know whether it is consistent, nor to use principled methods to maintain consistency [7]. Thus, contradictions are in this sense practically inevitable.

where T is the number of targets correctly identified,[6] A is the percentage of environmental area covered at the time the measurement is taken (this allows a measure of M to be taken at any time in the run, e.g., when $A = 0.25, A = 0.5, A = 0.75$ etc.), t is time elapsed, and P is the percentage of task completion (percentage of targets, out of all 100, correctly identified). Because a low performance time is generally only desirable when task completion is high, t is divided by P to penalize fast but sloppy performers.

In the case where the identification of the target is inference-based, and therefore liable to error (for instance, the agent has to tell the difference between lost hikers, park rangers, and large animals), tracking not just correct target IDs (True Positives, or TP) but also False Positives (FP), False Negatives (FN), and True Negatives (TN) will allow one to use the following standard performance metrics:

Sensitivity = $\frac{TP}{TP+FN}$

Specificity = $\frac{TN}{TN+FP}$

PPV (Positive Predictive Value) = $\frac{TP}{TP+FP}$

NPV (Negative Predictive Value) = $\frac{TN}{TN+FN}$

Although the bare metric M, and the measures for sensitivity, specificity, PPV and NPV, give one straightforward way to compute the performance of a given agent, and to compare the performance of different systems, when one is dealing with intelligent agents that can *learn*, it is also very important to measure the *change* in performance over time, and as a function of increased environmental complexity. Successive M values can be compared to assess the learning or improvement rate of the system. Likewise, successive values for the environmental complexity measures can be used to assess the agent's improving ability to handle increased environmental difficulty, for instance:

Ct (avg. complexity tolerance) = $\frac{\Delta C}{\Delta M}$

$V_o t$ (avg. variability tolerance) = $\frac{\Delta V_o}{\Delta M}$

$\delta_o t$ (avg. volatility tolerance) = $\frac{\Delta \delta_o}{\Delta M}$

$D_o t$ (avg. domain flexibility) = $\frac{\Delta D_o}{\Delta M}$

Similar metrics can of course be used for measuring changes in sensitivity, specificity, PPV, and NPV as a function of task complexity. These various measures taken together can give a clear picture of the perturbation tolerance of a given cognitive agent.

Finally, because the special abilities possessed by some intelligent agents, such as getting advice, reorganizing one's KB, or changing one's conceptual categories, can be very time-consuming, their worth depends a great deal on the value of accuracy -vs- the need for quickness in a given task. Thus in many cases it is sensible to introduce

[6]The variable T might also be calculated as correct IDs minus incorrect IDs ($TP - FP$, see below).

the domain variable R_V, a subjective measure of the importance of accuracy in the current task-domain. Although the variable R_V does not actually change anything about the domain itself, it can be used to inform the agent about the characteristics of its task. For the autonomous agent with complex cognitive abilities, and the ability to measure and track its *own* performance, R_V can provide a threshold measure as to when (and when not) to stop and ponder.

5. IMPLEMENTATION AND APPLICATION

A general test domain—PWorld—allowing for relatively easy characterization according to the suggested standard has been implemented as a component object model (COM) object on Microsoft Windows. PWorld is an $n \times n$ grid, and all elements of the world, including characterizing propositions, are stored and tracked in a database, with which PWorld communicates using ActiveX Data Objects (ADO). Active elements of the world—e.g. agents, weather, and such things as plants that can wither or grow—are implemented as separate COM objects that can communicate directly with the world, and indirectly with other active elements, by calling PWorld's various methods, such as: **addProposition()**, **sense()**, **move()**, and **eat()**.

PWorld was recently used to measure the perturbation tolerance of an agent using a standard reinforcement learning technique (Q-learning), and to compare it to the perturbation tolerance of an agent using a version of Q-learning that was enhanced with simple metacognitive monitoring and control (MCL) to create a *very simple* cognitive agent. The basic idea behind Q-learning is to try to determine which actions, taken from which states, lead to rewards for the agent (however these are defined), and which actions, from which states, lead to the states from which said rewards are available, and so on. The value of each action that could be taken in each state—its Q-value—is a time-discounted measure of the maximum reward available to the agent by following a path through state space of which the action in question is a part.

The Q-learning algorithm is guaranteed, in a static world, to eventually converge on an optimal policy [14], [15], regardless of the initial state of the Q-learning policy and the reward structure of the world. Moreover, if the world changes slowly, Q-learning is guaranteed to converge on near-optimal policies [16]. This is to say that Q-learners are already somewhat perturbation tolerant. However, we found that the actual performance of a Q-learner in the face of perturbation varies considerably, and, indeed, that post-perturbation performance is negatively correlated to the degree of perturbation ($R = -0.85, p < 0.01$). We further discovered that adding even a very simple metacognitive monitoring and control (MCL) component, that monitored reward expectations and, if expectations were repeatedly violated, instructed the Q-learner to change its policy in one of a number of ways, could greatly improve the perturbation tolerance of a Q-learner. The comparative performance results are summarized in Figure 1. The results show a high degree of correlation between the degree of the perturbation and the ratio of MCL to non-MCL performance ($R = 0.79, p < 0.01$). See [17] for details.

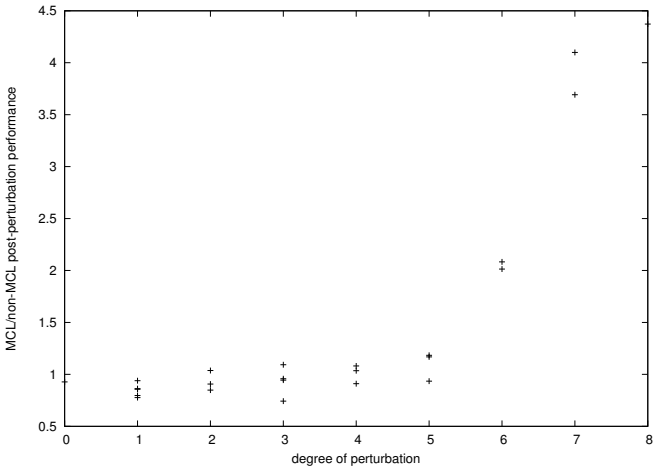

Fig. 1. Ratio of MCL/non-MCL post-perturbation performance, as a function of the degree of perturbation. ($R = 0.79, p < 0.01$)

However, from the standpoint of the current paper, what is important is the evaluation scheme in general, and our estimate of the "degree of perturbation" in particular. For this, the experiment must be understood in some more detail. To establish the above results, we built a standard Q-learner, and, starting with no policy (all Q-values=0), placed the Q-learner in an 8x8 grid-world—the possible states being locations in the grid—with reward r1 in square (1,1) and reward r2 in square (8,8). The initial reward structure [r1,r2] of the world was one of the following: [10,-10]; [25,5]; [35,15]; [19,21]; [15,35]; [5,25]. The Q-learner was allowed to take 10,000 actions in this initial world, which was enough in all cases to establish a very good albeit non-optimal policy. After receiving a reward, the Q-learner was randomly assigned to one of the non-reward-bearing squares in the grid. In turn 10,001, the reward structure was abruptly switched to one of the following: [25,5]; [35,15]; [19,21][7]; [15,35]; [5,25], [-10,10].

Our task-based performance measure for the Q-learner was the ratio of actual average reward per action taken (henceforth, per turn) to the ideal average reward per turn,

[7]Except when the initial structure was [19,21], in which case the post-perturbation structure was [21,19]

i.e., the average reward per turn theoretically available to a Q-learner following an optimal policy in the given environment. To get a handle on the difficulty of each perturbation, we first considered that the learned Q-table can be visualized as a topographic *overlay* on the grid world, where positive rewards are attractors, and negative rewards are repulsors, and the grade of the topography (the differences in the Q-values for each action at each location) corresponds to the degree of attraction to a given reward. Following the policy recommended by the Q-table is equivalent to moving downhill as quickly as possible. For simplicity, we can abstract considerably from this picture, and imagine that each square of the policy-overlay contains a proposition indicating the direction of the slope—toward (1,1), or toward (8,8). For a given perturbation, then, we can get one factor in the difficulty of the change, by counting the number of squares where the propositions characterizing the slope (as determined by an ideal policy) have changed. Thus, for instance, to go from the ideal abstract policy for reward structure [10,-10] (every square says go to (1,1)) to the abstract policy for reward structure [-10,10] (every square says go to (8,8)) involves a large *overlay difference* (D_o) of value 64, but going from [19,21] to [21,19] involves essentially no overlay difference.[8]

Another factor in measuring the degree of perturbation we considered for the current case was any valence change in the rewards. A valence change makes the perturbation greater because it makes it harder for the agent to actually change its abstract policy (one way to think about this might be as the mathematical equivalent of a contradiction). For instance, a negative reward that becomes positive (V^+) is masked from the agent because the policy is strongly biased against visiting that state. Thus, in light of the above considerations, we devised an equation to estimate the degree of perturbation (D_p) in each of the 22 cases:

$$D_p = D_o/16 + 3V^+ + V^- \tag{7}$$

The experiment as described primarily evaluated the perturbation tolerance of the agent in terms of its ability to move effectively between different (abstract) overlays, making the overlay difference the most relevant measure. However, other aspects of the test domain can indeed be measured according to the metrics offered here.

[8]It should be noted that this is an *adaptation* of the meaning of overlay and overlay difference to fit the experimental circumstances, and the nature of the agent being tested. If we understand the task of a Q-learner in terms of uncovering and mapping the reward-based topography of a given region, then this is the relevant difference between two regions that needs measuring when assessing the difficulty of moving from one to the other. Such adaptation of shared definitions and terms to individual circumstances is inevitable, and care must be taken in each case to properly explain individualized uses, and to remain sensitive to the overall goal of allowing cross-experiment comparisons.

n (overlay size) = 64. There are 64 squares in the overlay.

ρ_I (information density)= 3. Three propositions characterize each square: an X value and Y value that correspond to its location, and an R value that corresponds to the reward available there.

V_o (variability)= 0.36. The average minimum graph distance between squares in the grid is 5.5, and the average propositional difference is just above 2 (a square can differ by at most 3 propositions (X, Y and R), however most of the squares differ by 2 (X and Y, X and R, or Y and R), and a few by only 1 (X or Y)).

δ_o (volatility)= 0. The overlay does not change over time.

I (inconsistency)= $0\%/3\%$. Two values are given here, because when the agent begins the experiment, it has no beliefs, and there is therefore no inconsistency. However, when it moves between the two overlays, it has 64 beliefs about the rewards available in each square. Two of these beliefs are in direct conflict with the state of the world ($2/64 = 0.03$). Note the agent also has a number of beliefs about what actions to take in what circumstances to achieve maximum reward; many of these beliefs are false in its new circumstances. However they are not directly about the world, and nothing that the agent can perceive about the world *directly* contradicts any of these beliefs. Therefore, these do not count toward the measure of inconsistency.

U (uncertainty) = 0. The agent had perfect knowledge of its environment.

6. CHALLENGES

As the suggestions I have made are just that—preliminary suggestions meant as the starting point of a potentially long but important investigation, there remain some significant questions and challenges. First, and most obvious: are the elements of the environment identified here in fact the most important? And are the methods suggested for measuring them appropriate? Related to this: how easy will it be in practice to interpret a given test domain according to this proposal? For it is clear that even in the case where a domain is grid-available, and where it is therefore *possible* to apply these metrics, it will not necessarily be easy to do so. Although applying these metrics will be quite easy in domains like the one described above, where the parts and their contents are well defined, and even expressed in terms of the defined partition, it will be much less easy in test environments not designed along this model, for instance video games. Thus, some attention must be paid to developing principled, automated methods for analyzing test domains in accordance with the suggestions outlined here.

7. CONCLUSION

In this paper I have suggested a standard way to characterize the size, information density, variability, volatility, inconsistency and uncertainty of a given test environment, each of which contribute to the complexity of that environment. I have also suggested a way to measure the difference between two different environments of the same size. From these basic measures, I have shown how one can construct more comprehensive measures of the complexity of the environment, and I have given several examples of how the metrics can be used to measure the task performance and perturbation tolerance of cognitive agents. Finally, I showed how some of the metrics were applied to demonstrate that a metacognitive monitoring and control component could enhance the perturbation tolerance of a simple machine-learner. Although significant challenges remain, it is hoped that the paper will prove a useful starting point to the investigation of an important topic.

8. ACKNOWLEDGMENTS

This paper is a revised version of [18]. The research is supported in part by the AFOSR.

9. REFERENCES

[1] J. McCarthy, "Elaboration tolerance," in *Proceedings of the Fourth Symposium on Logical Formalizations of Commonsense Reasoning*, 1998.

[2] E. Amir, "Toward a formalization of elaboration tolerance: Adding and deleting axioms," in *Frontiers of Belief Revision*, M. Williams and H. Rott, Eds. Kluwer, 2000.

[3] M. Greenberg and D. Westbrook, "The phoenix testbed," 1990, technical Report UM-CS-1990-019, Computer and Information Science, University of Massachusetts at Amherst.

[4] P. R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe, "Trial by fire: Understanding the design requirements for agents in compex environments," 1989, technical Report UM-CS-1990-061, Computer and Information Science, University of Massachusetts at Amherst.

[5] M. Pollack and M. Ringuette, "Introducing the tileworld: experimentally evaluating agent architectures," in *Proceedings of the Eighth National Conference on Artificial Intelligence*, T. Dietterich and W. Swartout, Eds. Menlo Park, CA: AAAI Press, 1990, pp. 183–189. [Online]. Available: citeseer.nj.nec.com/pollack90introducing.html

[6] E. Durfee and T. Montgomery, "MICE: A flexible testbed for intelligent coordination experiments," in *Proceedings of the Ninth Workshop on Distributed AI*, Rosario, Washington, 1989, pp. 25–40. [Online]. Available: citeseer.nj.nec.com/durfee89mice.html

[7] D. Perlis, "On the consistency of commonsense reasoning," *Computational Intelligence*, vol. 2, pp. 180–190, 1986. [Online]. Available: http://www.cs.umd.edu/projects/active/doc/papers/86/occr.pdf

[8] K. Purang, "Systems that detect and repair their own mistakes," Ph.D. dissertation, Department of Computer Science, University of Maryland, College Park, Maryland, 2001.

[9] J. Elgot-Drapkin, "Step-logic: Reasoning situated in time," Ph.D. dissertation, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

[10] J. Elgot-Drapkin, S. Kraus, M. Miller, M. Nirkhe, and D. Perlis, "Active logics: A unified formal approach to episodic reasoning," Univ of Maryland, UMIACS and CSD, Tech. Rep. UMIACS TR # 99-65, CS-TR # 4072, 1993.

[11] J. Elgot-Drapkin and D. Perlis, "Reasoning situated in time I: Basic concepts," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 2, no. 1, pp. 75–98, 1990.

[12] K. Purang, D. Purushothaman, D. Traum, C. Andersen, and D. Perlis, "Practical reasoning and plan execution with active logic," in *IJCAI-99 Workshop on Practical Reasoning and Rationality*, 1999.

[13] M. Bhatia, P. Chi, W. Chong, D. P. Josyula, M. Anderson, Y. Okamoto, D. Perlis, and K. Purang, "Handling uncertainty with active logic," in *Proceedings of the AAAI Fall Symposium on Uncertainty in Computation*, 2001. [Online]. Available: http://www.cs.umd.edu/ mikeoda/papers/aaai01.pdf

[14] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, Cambridge, England, 1989.

[15] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[16] I. Szita, B. Takács, and A. Lörincz, "ϵ-MDPs: Learning in varying environments," *Journal of Machine Learning Research*, vol. 3, pp. 145–174, 2002.

[17] M. L. Anderson, T. Oates, W. Chong, and D. Perlis, "Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance," submitted.

[18] M. L. Anderson, "Specification of a test environment and performance measures for perturbation-tolerant cognitive agents," in *Proceedings of the AAAI Workshop on Intelligent Agent Architectures*, 2004. [Online]. Available: http://www.cs.umd.edu/ anderson/papers/aaai_metrics_04.pdf